**Department of Electrical Engineering**

|  |  |
| --- | --- |
| **Faculty Member: Mr. Kalim Ullah** | **Dated: 18th, February, 2018** |
|  |  |
| **Course/Section: BEE-8D** | **Semester: 6th** |
|  |  |

**EE330-304 Digital Signal Processing**

**Lab #4: Introduction to DSP Kit TMS 320C6713 DSK**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **PLO4** | | **PLO5** | **PLO8** | **PLO9** |
| **Name** | **Reg. No** | **Viva / Quiz / Lab Performance** | **Analysis of data in Lab Report** | **Modern Tool Usage** | **Ethics and Safety** | **Individual and Team Work** |
|  |  | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** |
| Muhammad Haris Rehman | 182797 |  |  |  |  |  |
| Muhammad Anus Siddiqui | 174308 |  |  |  |  |  |
| Muhammad Hasnain Afridi | 192414 |  |  |  |  |  |
|  |  |  |  |  |  |  |

# **Objectives**

The objective of this lab is to introduce the DSP Starter Kit C6713

* Getting started with DSP Kit
* Getting Started with Code Composer Studio
* Basic Code Compilation on DSP Kit
* Working with basic sinusoids on DSP Kit

# **Introduction:**

The hardware experiments in the DSP lab are carried out on the Texas Instruments TMS320C6713 DSP Starter Kit (DSK), based on the TMS320C6713 floating point DSP running at 225 MHz. The basic clock cycle instruction time is 1/(225 MHz)= 4.44 nanoseconds. During each clock cycle, up to eight instructions can be carried out in parallel, achieving up to 8×225 = 1800 million instructions per second (MIPS). The C6713 processor has 256KB of internal memory, and can potentially address 4GB of external memory.

The DSK board includes a 16MB SDRAM memory and a 512KB Flash ROM. It has an on-board 16-bit audio stereo codec (the Texas Instruments AIC23B) that serves both as an A/D and a D/A converter. There are four 3.5 mm audio jacks for microphone and stereo line input, and speaker and head-phone outputs. The AIC23 codec can be programmed to sample audio inputs at the following sampling rates:

fs = 8, 16, 24, 32, 44.1, 48, 96 kHz

The DSK also has four user-programmable DIP switches and four LEDs that can be used to control and monitor programs running on the DSP. All features of the DSK are managed by the CCS, which is a complete integrated development environment (IDE) that includes an optimizing C/C++ compiler, assembler, linker, debugger, and program loader.

The CCS communicates with the DSK via a USB connection to a PC. In addition to facilitating all programming aspects of the C6713 DSP, the CCS can also read signals stored on the DSP’s memory, or the SDRAM, and plot them in the time or frequency domains.

# **TMS 320 C 6713 Digital Signal Processor**

The TMS320C6713 (C6713) is based on the very long instruction word (VLIW) architecture, which is very well suited for numerically intensive algorithms. The internal program memory is structured so that a total of eight instructions can be fetched every cycle. For example, with a clock rate of 225 MHz, the C6713 is capable of fetching eight 32 - bit instructions every 1/(225 MHz) or 4.44 ns. Features of the C6713 include 264 kB of internal memory (8 kB as L1P and L1D Cache and 256 kB as L2 memory shared between program and data space), eight functional or execution units composed of six ALUs and two multiplier units, a 32 - bit address bus to address 4 GB (gigabytes), and two sets of 32 - bit general – purpose registers.

The C67xx processors (such as the C6701, C6711, and C6713) belong to the family of the C6x floating - point processors; whereas the C62xx and C64xx belong to the family of the C6x fixed - point processors. The C6713 is capable of both fixed and floating point processing.





# **Code Composer Studio**

Code Composer Studio (CCS) provides an integrated development environment (IDE) for real - time digital signal processing applications based on the C programming language. It incorporates a C compiler, an assembler, and a linker. It has graphical capabilities and supports real - time debugging. The C compiler compiles a C source program with extension .c to produce an assembly source file with extension *.asm* .

The assembler assembles an *.asm* source file to produce a machine language object file with extension *.obj* . The linker combines object files and object libraries as input to produce an executable file with extension *.out* . This executable file represents a linked common object file format (COFF), popular in Unix - based systems and adopted by several makers of digital signal processors [44] . This executable file can be loaded and run directly on the digital signal processor.

A Code Composer Studio project comprises all of the files (or links to all of the files) required in order to generate an executable file. A variety of options enabling files of different types to be added to or removed from a project are provided. In addition, a Code Composer Studio project contains information about exactly how files are to be used in order to generate an executable file. Compiler/linker options can be specified.

A number of debugging features are available, including setting breakpoints and watching variables, viewing memory, registers, and mixed C and assembly code, graphing results, and monitoring execution time. One can step through a program in different ways (step into, or over, or out). Real - time analysis can be performed using CCS’ s real - time data exchange(RTDX) facility. This allows for data exchange between the host PC and the target DSK as well as analysis in real - time without halting the target.

# **Task 1**

## **C Language Program:**

//sine8\_LED.c sine generation with DIP switch control

#include "dsk6713\_aic23.h" //codec support

Uint32 fs = DSK6713\_AIC23\_FREQ\_8KHZ; //set sampling rate

#define DSK6713\_AIC23\_INPUT\_MIC 0x0015

#define DSK6713\_AIC23\_INPUT\_LINE 0x0011

Uint16 inputsource=DSK6713\_AIC23\_INPUT\_MIC;//select input

#define LOOPLENGTH 8

short loopindex = 0; //table index

short gain = 10; //gain factor

short sine\_table[LOOPLENGTH]={0,707,1000,707,0,-707,-1000,-707}; //sine values

void main()

{

comm\_poll(); //init DSK,codec,McBSP

DSK6713\_LED\_init(); //init LED from BSL

DSK6713\_DIP\_init(); //init DIP from BSL

while(1) //infinite loop

{

if(DSK6713\_DIP\_get(0)==0) //=0 if DIP switch #0 pressed

{

DSK6713\_LED\_on(); //turn LED #0 ON

output\_left\_sample(sine\_table[loopindex++]\*gain); //output sample

if (loopindex >= LOOPLENGTH) loopindex = 0; //reset table index

}

else DSK6713\_LED\_off(0); //turn LED off if not pressed

} //end of while(1) infinite loop

} //end of main

## **Program Discription:**

An array, sine\_table , of eight 16 - bit signed integers is declared and initialized to contain eight samples of exactly one cycle of a sinusoid. The value of sine\_table[i] is equal to

*1000sin(2πi/8) for i 🡪1, 2, 3, . . . , 7*

Within function *main()* , calls to functions *comm\_poll()* , *DSK6713\_LED\_init()* , and *DSK6713\_DIP\_init()* initialize the DSK, the AIC23 codec onboard the DSK, and the two multichannel buffered serial ports (McBSPs) on the C6713 processor. Function *comm\_poll()* is defined in the file *c6713dskinit.c* , and functions *DSK6713\_LED\_init()* and *DSK6713\_DIP\_init()* are supplied in the board support library (BSL) file dsk6713bsl.lib .

The program statement while (1) within the function *main ()* creates an infinite loop. Within that loop, the state of DIP switch #0 is tested and if it is pressed down, LED #0 is switched on and a sample from the lookup table is output.

If DIP switch #0 is not pressed down then LED #0 is switched off. As long as DIP switch #0 is pressed down, sample values read from the array sine\_table will be output and a sinusoidal analog output waveform will be generated via the left - hand channel of the AIC23 codec and the LINE OUT and HEADPHONE sockets.

Each time a sample value is read from the array sine\_table , multiplied by the value of the variable gain , and written to the codec, the index, loopindex , into the array is incremented and when its value exceeds the allowable range for the array( LOOPLENGTH - 1 ), it is reset to zero.

## **Creating a Project:**

The given Program as mentioned above, was to be compiled and run on the Digital Signal processor. For this, we used CCSv6 and basically, our first task was to make a project in CCS and run it. To complete this task, all the steps were given and we followed the steps to create project, run our program and implement it in hardware.

# **Task 2**

This task is about the changing the frequency of the sinusoid which can be done in two ways as mentioned below;

## **Changing Sampling Frequency:**

We change the AIC23 codec sampling frequency from 8 kHz to 16 kHz by changing the line that reads;

Uint32 fs = DSK6713\_AIC23\_FREQ\_8KHZ;to readUint32 fs = DSK6713\_AIC23\_FREQ\_16KHZ;

We rebuilt (use incremental build) the project, loaded and ran the new executable file, and verified that the frequency of the generated sinusoid is 2 kHz.

The frequencies supported by the AIC23 codec are 8, 16, 24, 32, 44.1, 48, and 96 kHz.

## **Change Lookup Table:**

Secondly, to increase the frequency, we can change the number of samples stored in the lookup table to four. By changing the lines that read

#define LOOPLENGTH 8

short sine\_table[LOOPLENGTH]={0,707,1000,707,0, - 707,0, - 1000,- 707};

to read

#define LOOPLENGTH 4

short sine\_table[LOOPLENGTH]={0,1000,0, - 1000};

After implementation, we verified that the frequency of the sinusoid generated was 2 kHz (assuming an 8-kHz sampling frequency). The sinusoid is no longer generated if the DIP switch #0 is not pressed down.

A different DIP switch can be used to control whether or not a sinusoid is generated by changing the value of the parameter passed to the functions DSK6713\_DIP\_get(), DSK6713\_LED\_on(), *and* DSK6713\_ LED\_off(). Suitable values are 0, 1, 2, and 3. Two sliders can readily be used, one to change the gain and the other to change the frequency.

A different signal frequency can be generated, by changing the incremental changes applied to the value of loopindex within the C program (e.g., stepping through every two points in the table).

# **Conclusions**

In this lab, we were introduced to the DSP Starter Kit C6713. In this lab, we learnt to make project in CCSv6 (Code Composer Studio Version 6). This was a long procedure, but we had an experience how to program a c language program to a physical hardware.

All steps were mentioned in the lab and we just followed to complete our lab. Secondly, in the task 2 of this lab, the C program was given to us and we had to implement it. Secondly we had to change the frequency of the sinusoid which was produced by this program.